

Implementation of Privacy-preserving SimRank over Distributed Information Network

Yu-Wei Chu¹, Chih-Hua Tai², Ming-Syan Chen^{1,3} and Philip S. Yu⁴

¹Dept. of EE, National Taiwan University, Taipei, Taiwan

²Dept. of CSIE, National Taipei University, New Taipei, Taiwan

³Research Center of Info. Tech. Innovation, Academia Sinica, Taipei, Taiwan

⁴Dept. of CS, University of Illinois at Chicago, IL, USA

hanatai@mail.ntpu.edu.tw; mschen@citi.sinica.edu.tw; psyu@cs.uic.edu

Abstract

Information network analysis has drawn a lot attention in recent years. Among all the aspects of network analysis, similarity measure of nodes has been shown useful in many applications, such as clustering, link prediction and community identification, to name a few. As linkage data in a large network is inherently sparse, it is noted that collecting more data can improve the quality of similarity measure. This gives different parties a motivation to cooperate. In this paper, we address the problem of link-based similarity measure of nodes in an information network distributed over different parties. Concerning the data privacy, we propose a privacy-preserving SimRank protocol based on fully-homomorphic encryption to provide cryptographic protection for the links.

Index Terms

Privacy; Similarity;

I. INTRODUCTION

Real network data usually consist of objects of different types, forming a so called heterogeneous network. A heterogeneous network contains abundance of information, which gives rise to a great interest in its analysis such as clustering, classification, centrality measure, object ranking and pattern mining [4, 7, 14], to name a few. Among all the aspects of heterogeneous network analysis, measuring the similarity between nodes is one of the most important problems. Answering how similar nodes are is essential in many applications, such as clustering, link prediction and community identification [1, 5, 12].

According to the evaluation of [13], link-based similarity measures [6, 9, 10] well conform with human judgement. In the work of [10], the similarity of two nodes can be understood as a weighted count of the number of all-length paths between the nodes. In [9], the similarity can be regarded as the probability that a node A reaches a node B in a restricted random walk manner. Both the measures proposed in [10] and [9] are applied on undirected networks.

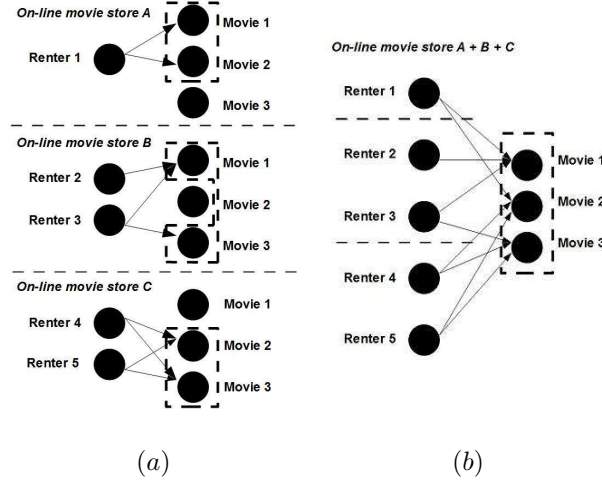


Fig. 1. The rental data of on-line movie stores A, B and C.

SimRank [6] defines the similarity score of two nodes A and B as the expected meeting distance, i.e., the expected distance that two random surfers, starting at node A and node B respectively, travel before they meet at the same node. Among all the link-based similarity measures, SimRank[6] is one of the most influential approaches [11] for following reasons. First, SimRank similarity score is independent to the clustering approaches, and thus can be applied to various kinds of clustering methods. Second, SimRank can be applied on both directed and undirected graphs, which give a great flexibility in analyzing heterogeneous networks. Furthermore, SimRank can work on any graphs that have object-to-object relations, without the need of domain knowledge.

In reality, however, data are often distributed among different parties. For example, Figure 1(a) shows the rental data in three on-line movie stores. Consider that an on-line movie store wants to build a recommendation system for its customers. Based on the rental records, SimRank can provide a good similarity measure for clustering the movies of similar types. However, the clustering result may still not be good enough since the amount of rental records is also a dominant factor. Combining the rental data from different stores for clustering, as shown in Figure 1(b), surely can improve the performance of movie recommendation systems. This gives the on-line stores a motivation to cooperate.

Whenever different parties cooperate, data privacy is an important concern. In the above example, the on-line movie stores may not be willing to reveal their network data to each other since the links indicating the rental information, i.e., customer-rent-movie, are usually one of the most valuable assets for a company. Revealing the rental information of a company may allow its competitor to be able to analyze its customer behavior and use the information against the company. Therefore, how to cooperatively compute SimRank similarity score without exposing any sensitive information, i.e., links between objects, becomes a crucial problem. To the best of our knowledge, there is not yet any studies addressing this problem.

It is challenging, under the constraint of link protection, to calculate the link-based similarity of nodes in a network distributed over different parties. As the similarity calculation is based on the link information of the entire

network, it is essential to securely integrate the distributed data into a synthesized graph. For example, according to the similarity definition of SimRank, two nodes are seen as similar if the neighbor nodes they connect to are themselves similar to each other. As the result, the entire graph structure needs to be considered when computing similarity scores between nodes. However, synthesizing graphs without exchanging any link information seems daunting. Without the knowledge of the entire graph structure, we can't even correctly identify the neighbors of a certain node. How to build a combined graph having access to the information we need and avoid revealing each party's link information in the same time is an issue needed to be solved.

Specifically, in this paper we take the link-based similarity measure defined by SimRank [6] and address the problem of privacy-preserving similarity measure in a joint bipartite network consisting of data from two or more parties. For such a problem, it has the challenges on (1) how each party can guarantee its link protection in a way that allows the construction of a structure containing the information involved for similarity measure, and (2) how the similarity is calculated on such a structure. To tackle these challenges, we propose a new protocol called privacy-preserving SimRank (PP-SimRank) based on fully-homomorphic encryption FHE [2, 3]. FHE takes only 1 or 0 as inputs, outputs different ciphertexts with random noise introduced, and has both the addition and multiplication homomorphic properties. Therefore, the XOR- and AND- operations on a plaintext are equivalent to the addition and multiplication on the corresponding ciphertext. PP-SimRank then utilizes these properties to construct *virtual* joint networks in ciphertext field. By extending the similarity definition to the virtual networks, PP-SimRank is able to calculate SimRank scores without knowing the link information in the bipartite network of each party. We show that PP-SimRank can protect the link information in a semi-honest model, where a semi-honest security model restricts all parties to faithfully follow their specified protocol, but allows the parties to record all the intermediate messages for analysis. In addition, to overcome the constraint of FHE taking only 1 or 0 as inputs, PP-SimRank converts the SimRank scores into binary representations and carries out the score computation in ciphertext field with binary arithmetic operations. That is, we demonstrate an application of FHE in practice.

Generally, the contributions of this paper include:

- 1) We are the first to address the problem of link-based similarity measure co-computation over distributed information network.
- 2) We propose a privacy-preserving protocol, PP-SimRank, to securely compute SimRank similarity score over distributed data. PP-SimRank is the first privacy-preserving protocol that focuses on link-based similarity measure co-computation.
- 3) We carry out the implementation of basic arithmetic operations, i.e., addition, subtraction, multiplication and division, in the ciphertext field under fully-homomorphic encryption, demonstrate an application of fully-homomorphic encryption scheme and show its potential in privacy-preserving data mining.

The rest of this paper is organized as follows. Section II formally defines our problem and provides the background knowledge to our work. Section III describes our Privacy-Preserving SimRank protocol in details, and Section IV discusses the implementation issues. We then analyze the complexity and security of the proposed PP-SimRank in

Section V, extend PP-SimRank to multi-party scenario in Section VI, and conclude this paper in Section VII.

II. PRELIMINARY

In this section, we formally define the problem and provide the necessary background. We first formulate the problem in Section II-A, and give a brief review of SimRank in Section II-B.

A. Problem Formulation

In this paper, we address the problem of privacy-preserving computation of link-based similarity measure over horizontally partitioned information network in different parties. Our target is to protect the link information and achieve the computation of SimRank score simultaneously. For simplicity, we first focus on two-party scenario, and then show the extension to general cases.

In a two-party scenario, two parties called Alice and Bob hold information networks $G^A = (V^A, U, E^A)$ and $G^B = (V^B, U, E^B)$, respectively. Without loss of generality, we assume that V^A and V^B are disjoint vertex sets, and U is a vertex set known to both Alice and Bob. The parties wish to cooperatively learn the SimRank similarity scores $S(u_i, u_j)$ of all vertex pairs $\{u_i, u_j\} \in U$ using their joint data, i.e., $G = (V^A \cup V^B, U, E^A \cup E^B)$. For privacy concern, the problem is how to construct the virtual information network G without revealing E^A and E^B and how to perform the SimRank computation on the virtual information network G .

For example, consider that Alice and Bob are two on-line movie stores. Alice and Bob have different customer groups, V^A and V^B , respectively, and the same movie set U . A rental record is then a link connecting v_i with u_j . When both Alice and Bob regard their own rental records as sensitive data, the problem is that, how Alice and Bob can construct a virtual graph $G = (V^A \cup V^B, U, E^A \cup E^B)$ without revealing $E^A = \{(v_i^A, u_j)\}$ and $E^B = \{(v_i^B, u_j)\}$ to each other, and learn the SimRank similarity score $S(u_i, u_j)$ of all the movie pairs to improve their movie recommendation system.

For this problem, ideally, Alice and Bob would find a trusted third party to perform all the needed calculations for them. That is, Alice and Bob would securely transmit their data to this trusted third party. Let the third party compute the movies' pairwise similarity scores using SimRank and send the result information back to both Alice and Bob. However, such independent third parties are very hard to find or do not exist in the real world.

Instead of counting on a third party, we proposed a privacy-preserving protocol, PP-SimRank, based on fully-homomorphic encryption (FHE) [2, 3] to protect the link information. The fully-homomorphic encryption method takes the plaintexts of binary form, i.e., $\{0, 1\}$, as inputs and outputs the ciphertexts in the form of big integers. Given the same plaintext and the same public key for encryption, FHE will output different ciphertext (big integer) with random noise, i.e., there is a negligibly small probability [3, 8] that $\text{Enc}(1) = \text{Enc}(1)$. Therefore, FHE is semantic secure against chosen plaintext attacks and is suitable for link encryption. In addition, FHE has the following two properties.

PROPERTY 1. Addition-homomorphic

$$c_1 + c_2 = Enc(m_1) + Enc(m_2) = Enc(m_1 \oplus m_2), \quad (1)$$

PROPERTY 2. Multiplication-homomorphic:

$$c_1 \times c_2 = Enc(m_1) \times Enc(m_2) = Enc(m_1 \wedge m_2), \quad (2)$$

where the two ciphertexts $c_1 = Enc(m_1)$ and $c_2 = Enc(m_2)$ represent the encryption outcomes of the plaintexts m_1 and m_2 respectively. Accordingly, the addition and multiplication operations on the ciphertexts are equivalent to the XOR- and AND-operations on the corresponding plaintexts. We can, therefore, carry out arithmetic operations such as addition, subtraction, multiplication and division in the ciphertext field, using the homomorphic properties of FHE [2, 3]. By restricting the computation of SimRank similarity score in the ciphertext field, we can achieve the protection of link information.

Further, we will show that PP-SimRank can keep the link information remain hidden in a semi-honest security model.

B. SimRank Overview

SimRank [6] is a similarity measure based on the linkage information in a network. Two objects are regarded as similar ones if they are related to similar objects. Specifically, let $S(v_i, v_j)$ denote the similarity between two nodes v_i and v_j in a network $G = (V, E)$. The iterative similarity computation equation of SimRank is as follows:

$$S(v_i, v_j) = \frac{d_{out}}{|O(v_i)||O(v_j)|} \sum_{k=1}^{|O(v_i)|} \sum_{l=1}^{|O(v_j)|} S(O_k(v_i), O_l(v_j)), \quad (3)$$

$$S(u_i, u_j) = \frac{d_{in}}{|I(u_i)||I(u_j)|} \sum_{k=1}^{|I(u_i)|} \sum_{l=1}^{|I(u_j)|} S(I_k(u_i), I_l(u_j)), \quad (4)$$

where $O(v_i)$ or $O(v_j)$ denote the set of out-neighbors of nodes v_i or v_j , $I(u_i)$ or $I(u_j)$ denote the set of in-neighbors of nodes u_i or u_j , and d_{out} (d_{in}) is the decay factor.

To simplify the SimRank score computation, SimRank derives a node-pair graph $G^2 = (V^2, E^2)$. Each node w in V^2 is associated with a node pair $\langle a, b \rangle$ in G , i.e., $a, b \in V$. The presence of an edge between two nodes $\langle a, b \rangle$ and $\langle c, d \rangle$ in G^2 represents that there exist edges (a, c) and (b, d) in G or, there exist edges (a, d) and (b, c) in G . In G^2 , each node w is associated with a SimRank score giving a measure of similarity between the two nodes of G represented by w . The neighbors of w are the nodes whose similarity is needed to be considered when the SimRank score of w is computed.

Figure 2(a) shows an example of the joint rental data of two on-line movie stores Alice and Bob. Figure 2(b) is the corresponding node-pair graph G^2 , where the isolated nodes are omitted. There is an edge in G^2 between

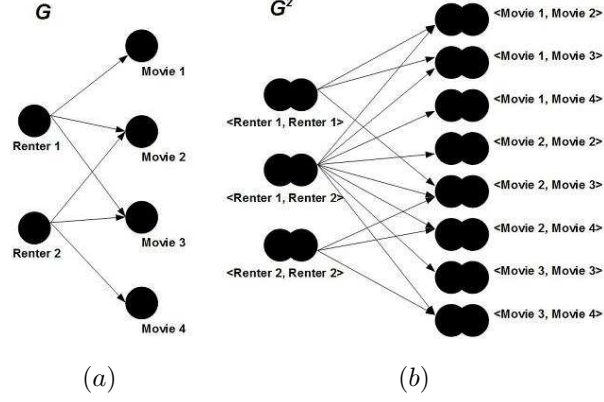


Fig. 2. The combined renter data of online movie stores Alice and Bob.

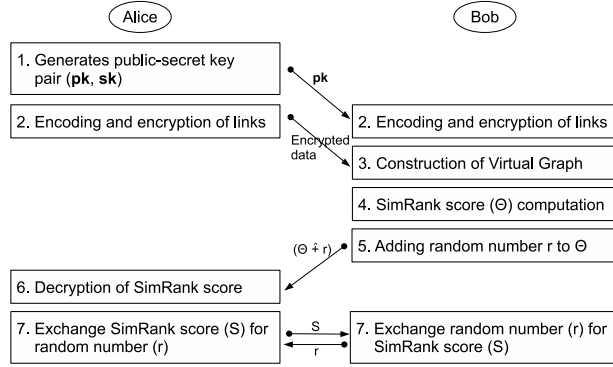


Fig. 3. Privacy-preserving SimRank protocol.

the nodes $\langle Movie1, Movie2 \rangle$ and $\langle Renter1, Renter2 \rangle$, because Renter 1 rents Movie 1 and Renter 2 rents Movie 2 in G .

In SimRank method, the node-pair graph G^2 is proposed to explicitly figure out the propagation of similarity scores between pairs of nodes. Later, we will further incorporate the node-pair graph G^2 into the definition of similarity so that PP-SimRank is able to calculate the SimRank score in ciphertext field.

III. PRIVACY-PRESERVING SIMRANK PROTOCOL

A. Protocol Overview

We now introduce a new protocol called Privacy-Preserving SimRank (PP-SimRank) based on fully-homomorphic encryption (FHE). In this protocol, there are two essential roles: cryptographer and calculator. Cryptographer determines the public-secret key pair (pk, sk) used for encryption/decryption of data, shares the public key with other while keeping the secret key strictly to himself, and performs the decryption of encrypted SimRank score. Calculator collects encrypted data from all parties, constructs the virtual network in the ciphertext field, and performs the SimRank score calculation on the virtual network. As cryptographer does not have the virtual network and calculator does not have the secret key for decryption, the SimRank score computation over distributed information

network can be achieved while the link information of each parties' data is protected.

Figure 3 shows the procedure of PP-SimRank protocol. First, one party, called Alice, determines the public-secret key pair $(\mathbf{pk}, \mathbf{sk})$, and sends the public key \mathbf{pk} to the other party, called Bob. After that, both Alice and Bob encode and encrypt their data with \mathbf{pk} , respectively, Alice then also transmits her encrypted data to Bob so that Bob will hold a joint virtual information network $G = (V^A \cup V^B, U, E^A \cup E^B)$ in the ciphertext field. Based on the joint network G , Bob further builds a node-pair graph G^2 . Later, by incorporating the virtual node-pair network G^2 into the definition of SimRank similarity, Bob performs the SimRank calculation in ciphertext field. Finally, Bob adds a random number to each of the final SimRank scores of the vertex pairs in U^2 , and sends the results to Alice for decryption. Both Alice and Bob can then learn the SimRank similarity scores of the vertex pairs in U^2 after they exchange the decrypted scores and the corresponding added random numbers.

In this protocol, however, there are several challenging problems needed to be solved. First, the bipartite network of each party has to be encoded and encrypted in a way that allows the constructions of the joint network G as well as the corresponding node-pair graph G^2 while the link protection is guaranteed. Second, based on the node-pair graph G^2 built in ciphertext field, the whole process of SimRank similarity computation has to be restricted in the ciphertext field as well, since the FHE homomorphic properties hold only if the plaintexts are encrypted with the same key. Moreover, it makes the problem even more challenging that FHE has the constraint of taking only 1 or 0 as inputs while the similarity score is a real number in $[0, 1]$. We will tackle these challenges step by step in the following subsections.

In Section III-B, we explain how Alice and Bob encrypt their data so that Bob is able to construct the graph matrix of the node-pair graph G^2 in the ciphertext field. Section III-C shows how exactly the graph matrix of G^2 is constructed. Based on G^2 , Section III-D gives the details of how the SimRank score computation is achieved, when the computation is restricted in the ciphertext field.

B. Encoding and encryption of links

We now explain how the parties encode and encrypt their data in details. Without loss of generality, we assume that both Alice and Bob know the total number of nodes in both U and $(V^A \cup V^B)$, i.e., $m = |V^A| + |V^B|$ and $n = |U|$, and all the nodes are ordered.

Note that the bipartite graph in our problem is not a weighted one. The direct relationship between two nodes is either connection or disconnection, i.e., a link $e \in \{0, 1\}$. In many cryptographic encryptions, it will output the same ciphertext given the same input and encryption key. As a result, the link protection still cannot be guaranteed even though the data is encrypted.

To tackle this challenge, we propose PP-SimRank based on FHE [2, 3], which takes only 1 or 0 as possible input and outputs different ciphertexts for a given input bit by introducing random noises in the encryption. That is, after encryption, the ciphertext indicating the same bit varies. To hide the link information more effectively, we let the parties encode not only the connections but also the disconnections between nodes into bit vectors and encrypt the bit vectors using FHE. For Bob who receives Alice's data, he thus cannot distinguish the ciphertexts corresponding

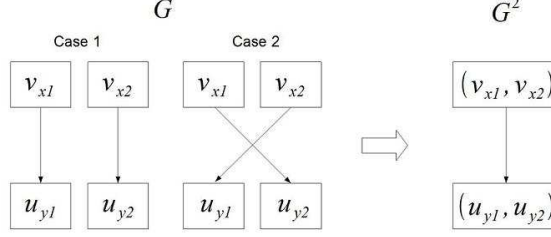


Fig. 4. Two cases lead to an edge between (v_{x1}, v_{x2}) and (u_{y1}, u_{y2}) .

case 1	case 2	case 1 \oplus case 2	case 1 \wedge case 2	$\epsilon_{x,y}$
1	1	0	1	1
1	0	1	0	1
0	1	1	0	1
0	0	0	0	0

Fig. 5. Or-equivalent operations.

to the 1- bits and 0- bits. The link information is effectively protected. On the other hand, as the link information is encoded into bits independently and encrypted without other artificial noise (in addition to the random noise introduced by FHE), Bob is able to utilize the homomorphic properties to construct the virtual graphs G and G^2 .

Specifically, for each node v_i in V^A (or V^B), Alice (or Bob) will generate a bit vector $E_{v_i}^A$ (or $E_{v_i}^B$), where $E_{v_i}^A, E_{v_i}^B \in \{1, 0\}^n$, with respect to the connections and disconnections of v_i to all the nodes in U based on her (his) own data. Each element $e_{i,j}$ in $E_{v_i}^A$ (or $E_{v_i}^B$) is 1 if there is a link in G^A (or G^B) connecting v_i and $u_j \in U$, or 0 otherwise. After encoding all the link information into bit vectors, the parties encrypt every element $e_{i,j}$ in the bit vectors $E_{v_i}^A$ and $E_{v_i}^B$ to get the cipher vectors $C_{v_i}^A$ and $C_{v_i}^B$, where each element $c_{i,j} = \text{Enc}(e_{i,j})$, using the same public key \mathbf{pk} (determined by Alice in the previous step). According to the homomorphic properties of FHE, Bob can thus collect the cipher vectors $C_{v_i}^A$ and $C_{v_i}^B$ to construct a virtual joint bipartite graph in the ciphertext field.

C. Construction of Virtual Graphs

This subsection shows how a joint virtual network G^2 , where $G = (V^A \cup V^B, U, E^A \cup E^B)$, is constructed in ciphertext field.

To construct the virtual network G^2 , knowing the link information is essential. However, since the link information is hidden in the cipher vectors $C_{v_i}^A$ and $C_{v_i}^B$, it is impossible for Bob to construct a graph with explicit link information. Instead, we let Bob construct the virtual graph G^2 by filling the graph matrix M of G^2 in ciphertext field. Each cipher element of M can be derived by applying the homomorphic operations on the cipher vectors $C_{v_i}^A$ and $C_{v_i}^B$ as follows.

Specifically, let $w_x = \langle v_{x1}, v_{x2} \rangle$ denotes a node-pair in G^2 , where v is a node in $(V^A \cup V^B)$, and $w_y = \langle$

$u_{x1}, u_{x2} >$ denotes the node-pair of U in G^2 . Recall that, in G^2 , there is an edge $\epsilon_{x,y}$ connecting the nodes $w_x = < v_{x1}, v_{x2} >$ and $w_y = < u_{y1}, u_{y2} >$ if and only if (as shown in Figure 4), in G , at least one of the two conditions in the following is satisfied.

- 1) $\{e_{x1,y1}, e_{x2,y2}\} \in (E^A \cup E^B)$: u_{y1} is an out-neighbor of v_{x1} and u_{y2} is an out-neighbor of v_{x2} .
- 2) $\{e_{x1,y2}, e_{x2,y1}\} \in (E^A \cup E^B)$: u_{y2} is an out-neighbor of v_{x1} and u_{y1} is an out-neighbor of v_{x2} .

That is, the value of $\epsilon_{x,y}$ can be derived by the following formula:

$$\epsilon_{x,y} = case1 \vee case2, \quad (5)$$

where

$$case1 = e_{x1,y1} \wedge e_{x2,y2},$$

$$case2 = e_{x1,y2} \wedge e_{x2,y1}.$$

When the OR-operation is carried out by XOR- and AND- operations, Equation (5) can be rewritten as follows based on Figure 5.

$$\epsilon_{x,y} = (case1 \oplus case2) \oplus (case1 \wedge case2). \quad (6)$$

Now consider this problem in ciphertext field and let $\pi_{x,y}$ be the corresponding cipher of $\epsilon_{x,y}$ in M . Due to the homomorphic properties of FHE, the XOR-/AND- operation on the plaintext is equivalent to the addition/multiplication operation on the corresponding ciphertext. The Value of $\pi_{x,y}$ can thus be derived as follows.

$$\begin{aligned} \pi_{x,y} &= Enc_{\mathbf{pk}}(\epsilon_{x,y}) \\ &= (Enc_{\mathbf{pk}}(case1) + Enc_{\mathbf{pk}}(case2)) \\ &\quad + (Enc_{\mathbf{pk}}(case1) \times Enc_{\mathbf{pk}}(case2)), \end{aligned} \quad (7)$$

where

$$Enc_{\mathbf{pk}}(case1) = c_{x1,y1} \times c_{x2,y2},$$

$$Enc_{\mathbf{pk}}(case2) = c_{x1,y2} \times c_{x2,y1}.$$

Consequently, Bob can construct a virtual graph G^2 for subsequent SimRank score computation, while the link information is protected and hidden in ciphers.

D. Computation of SimRank score

In this subsection, we explain in details that how Bob computes the SimRank score in ciphertext field, based on the virtual network G^2 . As Equations (3) and (4) show, the SimRank similarity of two nodes v_{x1} and v_{x2} is defined as the normalized sum of the similarities between their neighbors. In the ciphertext field, however, the neighborhood information as well as the link information is protected and hidden in the encrypted graph matrix M of G^2 . To compute the SimRank score, we then need to incorporate the graph matrix M into Equations (3) and (4), and express the formulas correspondingly in ciphertext field.

First, consider to incorporate the graph matrix M into Equations (3) and (4). For $v \in (V^A \cup V^B)$ and $u \in U$, let $w_x = \langle v_{x1}, v_{x2} \rangle$ and $w_y = \langle u_{y1}, u_{y2} \rangle$ denote the node-pairs in G^2 , and $\epsilon_{x,y}$ be the element in M that represents the connection/disconnection between the node-pairs w_x and w_y . The SimRank score associating with w_x and w_y is:

$$\begin{aligned}
 S(w_x) &= S(v_{x1}, v_{x2}) \\
 &= \frac{d_{out}}{|O(v_{x1})||O(v_{x2})|} \sum_{i=1}^{|O(v_{x1})|} \sum_{j=1}^{|O(v_{x2})|} S(O_i(v_{x1}), O_j(v_{x2})) \\
 &= \frac{d_{out}}{\sum_{i=1}^n e_{x1,i} \sum_{j=1}^n e_{x2,j}} \sum_{w_y \in G^2} S(w_y) \times \epsilon_{x,y},
 \end{aligned} \tag{8}$$

$$\begin{aligned}
 S(w_y) &= S(u_{y1}, u_{y2}) \\
 &= \frac{d_{in}}{|I(u_{y1})||I(u_{y2})|} \sum_{i=1}^{|I(u_{y1})|} \sum_{j=1}^{|I(u_{y2})|} S(I_i(u_{y1}), I_j(u_{y2})) \\
 &= \frac{d_{in}}{\sum_{i=1}^m e_{i,y1} \sum_{j=1}^m e_{j,y2}} \sum_{w_x \in G^2} S(w_x) \times \epsilon_{x,y}.
 \end{aligned} \tag{9}$$

In Equation (8), the $\sum_{i=1}^n e_{x1,i}$ ($\sum_{j=1}^n e_{x2,j}$) indicates the number of in-neighbors of v_{x1} (v_{x2}), and $S(w_y)$ is the SimRank similarity between v_{x1} 's potential neighbor and v_{x2} 's potential neighbor. Whether a $S(w_y)$ score will be counted in the $S(w_x)$ or not depends on the value of $\epsilon_{x,y} \in \{0, 1\}$ in the graph matrix M of G^2 .

Now we show the corresponding calculation of SimRank score in cipher field. The challenge is that, both the network G and the corresponding node-pair graph G^2 are built in ciphers. We then need to encrypt the SimRank similarity score using the same public key \mathbf{pk} , since the FHE homomorphic properties hold only when the score is encrypted in the same field as the link information.

Specifically, note that the FHE method takes only binary integer, i.e., $\{1/0\}$, as inputs, while the similarity score $S(w_x)$ is a real number $\in [0, 1]$. To allow the encryption of $S(w_x)$ using FHE, we encode the real number $S(w_x) \in [0, 1]$ (as well as d_{in} and d_{out}) into binary representation, i.e., a bit string, and encrypt the bit string into cipher string. The arithmetic operations on the real numbers then change to the binary arithmetic operations on the (bit/cipher) string. Let $\hat{\times}$, $\hat{\div}$, and $\hat{\sum}$ denote the multiplication, division and sum binary arithmetic operations, respectively. We discuss the implementations of the binary arithmetic operations in Section IV-A, and convert Equations (8) and (9) into the following formulas in ciphertext field.

$$\begin{aligned}\Theta(w_x) &= Enc_{\mathbf{pk}}(S(w_x)) \\ &= \Delta \hat{\times} \sum_{w_y \in G^2} (\Theta(w_y) \hat{\times} \pi_{x,y}),\end{aligned}\tag{10}$$

$$\begin{aligned}\Theta(w_y) &= Enc_{\mathbf{pk}}(S(w_y)) \\ &= \square \hat{\times} \sum_{w_x \in G^2} (\Theta(w_x) \hat{\times} \pi_{x,y}),\end{aligned}\tag{11}$$

where

$$\begin{aligned}\Delta &= Enc_{\mathbf{pk}}(d_{out}) \hat{\div} (\sum_{i=1}^n c_{x1,i} \hat{\times} \sum_{j=1}^n c_{x2,j}), \\ \square &= Enc_{\mathbf{pk}}(d_{in}) \hat{\div} (\sum_{i=1}^m c_{i,y1} \hat{\times} \sum_{j=1}^m c_{j,y2}).\end{aligned}$$

According to Equations (7), (10) and (11), Bob is able to perform the whole process of SimRank similarity calculation in ciphers without knowing the explicit link information of Alice's data. PP-SimRank, therefore, achieves a secure similarity measure on a joint networks consisting of different parties' data.

IV. IMPLEMENTATION ISSUES

A. Arithmetic operations in the ciphertext field

As mentioned in Section III-D, we implement the binary arithmetic operations to perform the addition, multiplication and division on the cipher strings representing $\Theta(w)$. First, we implement two virtual circuit functions, a full-adder function and a full-subtractor function, that can simulate the functionality of a real full-adder and full-subtractor circuit using the fully-homomorphic properties of FHE, respectively. By combining a sequence of the full-adder (or full-subtractor) function, we carry out a binary addition $\hat{+}$ (or subtraction $\hat{-}$) function that can perform the addition (or subtraction) of two cipher strings. After that, the multiplication function $\hat{\times}$ can be developed by calculating the partial products of the input cipher strings, shifting each of the product to the left, and adding them together using the addition function $\hat{+}$. Similarly, division function $\hat{\div}$ consists of subtraction function $\hat{-}$ and the shifting mechanism. We explain the details in the following.

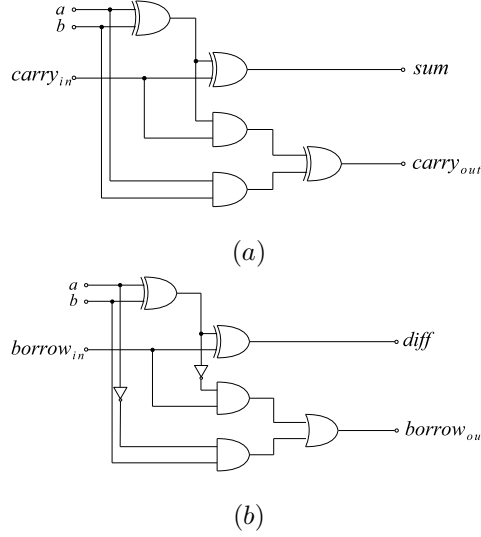


Fig. 6. Logical circuits of (a) a full-adder and (b) a full-subtractor.

Binary addition $\hat{+}$. As shown in Figure (6)(a), in electronics, a real full-adder is composed by a number of logical operations. The real full-adder adds the operand bits a and b with the carried-in bit $carry_{in}$ received from the previous full-adder, and output the answer bit sum and carry-out bit $carry_{out}$ as the result. The operations of a real full-adder can be expressed in terms of XOR and AND as follows:

$$\begin{aligned} sum &= a \oplus b \oplus carry_{in}, \\ carry_{out} &= (a \wedge b) \oplus (carry_{in} \wedge (a \oplus b)). \end{aligned} \quad (12)$$

As these operations are in bit level the same as the inputs of FHE, we then can convert the real full-adder into ciphertext field using the fully-homomorphic properties. By replacing the XOR- and AND- operations with addition and multiplication, respectively, Equation (12) is transformed into the following formulas:

$$\begin{aligned} c_{sum} &= c_a + c_b + c_{carry_{in}}, \\ c_{carry_{out}} &= (c_a \times c_b) + (c_{carry_{in}} \times (c_a + c_b)), \end{aligned} \quad (13)$$

where c_{sum} , c_a , c_b and c_{carry} denote the corresponding ciphers of sum , a , b and $carry$, respectively. Based on Equation (13), we then develop a function, called virtual full-adder function which executes the functionality of the real full-adder circuit in ciphertext field. After that, let each similarity score $S(w)$ in PP-SimRank use l bits for its binary representation. By connecting l virtual full-adder function together, i.e., each function takes the output cipher $c_{carry_{out}}$ of previous full-adder function as its input carry-in cipher $c_{carry_{in}}$, we can carry out a binary addition function $\hat{+}$ to add cipher strings of $\Theta(w)$ in ciphertext field.

Binary multiplication $\hat{\times}$. We further utilize the binary addition function $\hat{+}$ to implement the binary multiplication function $\hat{\times}$. Specifically, similar to what we usually do in the computation of multiplying two decimal numbers, the binary multiplication function $\hat{\times}$ can be carried out by computing a sequence of partial products, shifting the resulting partial products to the left, and finally applying the binary addition function $\hat{+}$ to add the products together.

Similarly, we begin from the real full-subtractor circuit and then explain the implementations of binary subtraction $\hat{-}$ and binary division $\hat{\div}$ operations in ciphertext field.

Binary subtraction $\hat{-}$. The logical operations of a real full-subtractor shown in Figure 6(b) can be expressed as follows:

$$\begin{aligned} diff &= a \oplus b \oplus borrow_{in}, \\ borrow_{out} &= (\bar{a} \wedge b) \vee (borrow_{in} \wedge \overline{(a \oplus b)}) \\ &= ((1 \oplus a) \wedge b) \vee (borrow_{in} \wedge (1 \oplus (a \oplus b))), \end{aligned} \tag{14}$$

where the NOT-operation on a bit a is achieved by computing the value of $(1 \oplus a)$. Let

$$\begin{aligned} t1 &= (1 \oplus a) \wedge b, \\ t2 &= borrow_{in} \wedge (1 \oplus (a \oplus b)). \end{aligned} \tag{15}$$

Based on the truth tables in Figure 5, $borrow_{out}$ can also be expressed in terms of XOR- and AND- logics.

$$borrow_{out} = (t1 \oplus t2) \oplus (t1 \wedge t2). \tag{16}$$

Accordingly, a virtual full-subtractor in ciphertext field is derived in Equation (17) (from Equations (14), (15) and (16)) again by substituting the XOR- and AND- logics with addition and multiplication, respectively.

$$\begin{aligned} c_{diff} &= c_a + c_b + c_{borrow_{in}}, \\ c_{borrow_{out}} &= (c_{t1} + c_{t2}) + (c_{t1} \times c_{t2}), \end{aligned} \tag{17}$$

where

$$\begin{aligned} c_{t1} &= (c_1 + c_a) \times c_b, \\ c_{t2} &= (c_{borrow_{in}} \times (c_1 + (c_a + c_b))). \end{aligned}$$

TABLE I

THE EXECUTION TIME OF FHE ENCRYPTION, FHE RE-ENCRYPTION AND CIPHERTABLE LOOK-UP W.R.T. VARIOUS SECURITY LEVELS.

Security Level (dimension)	FHE Encryption (sec)	FHE Re-Encryption (sec)	CipherTable Look-up (sec)
512	0.217	6.3	0.020
2048	2.084	34.1	0.254
8192	20.300	185	2.397

By regarding a virtual full-subtractor as a basic unit, we can thus implement a binary subtractor $\hat{-}$ in a similar way of building a binary addition $\hat{+}$, where each function takes the output cipher $c_{borrow_{out}}$ of previous full-subtractor function as its input borrow-in cipher $c_{borrow_{in}}$.

Binary division $\hat{\div}$. The binary division function $\hat{\div}$ also can be carried out by utilizing a sequence of binary subtraction function $\hat{-}$. Similar to the decimal division, the binary division function $\hat{\div}$ computes the quotient cipher string by iteratively subtracting divisor from the dividend using binary subtraction function $\hat{-}$ and shifting the divisor to the right after each binary subtraction $\hat{-}$. Each ciphertext in the quotient cipher string is determined by adding the output borrow cipher $c_{borrow_{out}}$ of the binary subtraction $\hat{-}$ with a cipher $c_1 = Enc_{pk}(1)$, which is equivalent to the NOT-operation ($1 \oplus borrow_{out}$) in the plaintext field.

B. Efficiency improvement

FHE encrypts bits into ciphertexts (with random noises) in the form of very big integers to achieve semantic security. A side effect, however, is that operations on big integers usually cost a lot of time. In this subsection, we propose to reduce the execution time of PP-SimRank in three ways: reducing the number of binary multiplications in the score calculations, replacing the FHE encryption with look-up, and carrying out the FHE re-encryption mechanism by Re-encryption protocol instead.

Reducing the number of binary multiplications. Equations (10) and (11) give the detailed calculations of SimRank similarity in ciphertext field. We note that the execution efficiency can be improved by replacing the binary multiplication $\hat{\times}$ of $(\Theta(w_x) \hat{\times} \pi)$ with simple multiplications \times . Specifically, recall that as mentioned in Section III-C, a ciphertext π represents a bit ϵ in the plaintext, and each SimRank score $\Theta(w_x)$ in the ciphertext field is represented in the form of a cipher string corresponding to the binary representation (a bit string) of $S(w_x)$ in plaintexts. Because ϵ is either 0 or 1, multiplying ϵ with the bit string of $S(w_x)$ can be carried out by AND-operations between them. Correspondingly, in the ciphertext field, we can achieve $(\Theta(w_x) \hat{\times} \pi)$ in Equations (10) and (11) by multiplying (corresponding to AND-operation in plaintext field) π with every ciphertext in the cipher string of $\Theta(w_x)$, instead of performing the time consuming binary multiplication.

CipherTable look-up. Table I shows the FHE encryption time of one bit with respect to different security levels¹.

¹In [3], they use lattices of 512, 2048 and 8192 dimensions to provide security levels called as toy, small and medium.

Accordingly, it would be infeasible if Alice and Bob encrypt their data by calling the FHE encryption function.

To solve this problem, we use CipherTable look-up instead which is at least 8 times faster than FHE encryption according to the results in Table I. That is, since there are only two possible inputs 1 and 0 of FHE, Alice and Bob can share the public key \mathbf{pk} in advance and respectively prepare a CipherTable containing a set of ciphertexts w.r.t. the plaintexts $\{0, 1\}$ before they really start computing the similarity measures on their joint data. Therefore, when PP-SimRank is performed, Alice and Bob can encrypt their data in step 2 of Figure 3 by simply choosing a corresponding ciphertext from their own CipherTable, instead of calling FHE encryption function. As FHE outputs different ciphertexts (by introducing a random noise in the encryption) and the CipherTable is built by each party itself, this replacement is satisfactory.

Re-encryption protocol. The PP-SimRank protocol protects the link information using FHE. For a given bit representing direct connection or disconnection, FHE will introduce a random noise into the encryption to output different ciphertexts. Due to the random noise, the inverse mapping from ciphers to the bit becomes challenging. However, the noise in a ciphertext will propagate and grow with the homomorphic operations operated on it. When the magnitude of noise reaches a certain limitation², the ciphertext will not be able to be correctly decrypted. A re-encryption mechanism that can refresh the ciphertext by reducing the associated noise is thus required.

The work in [3] has suggested a mechanism, called bootstrapping, for re-encryption. The idea of bootstrapping is to refresh a ciphertext by decrypting the ciphertext homomorphically with the encrypted secret key. As shown in Table I, however, the re-encryptions of bootstrapping under various security level are very time-consuming. Instead, we propose an more efficient re-encryption protocol for our problem scenario. The results in Table I shows that our re-encryption protocol runs at least 77 times faster than FHE re-encryption method.

Specifically, when it is detected³, by Bob, that the magnitude of noise associating with a ciphertext reaches a limitation during the computation, Bob asks Alice's help for re-encryption. The noisy ciphertext is sent to Alice. Alice then decrypts the noisy ciphertext using the secret key \mathbf{sk} (which is kept strictly to Alice), and encrypts the resulting bit again using the same public key \mathbf{pk} . Here, as mentioned previously, Alice can perform the encryption by looking up her CipherTable, rather than call the FHE encryption function. The re-encryption efficiency is thus improved significantly. After that, a corresponding new ciphertext is sent back to Bob for subsequent computation.

This re-encryption protocol will not damage the private link protection. For Bob, what he sends out and receives are both ciphertexts. Therefore, he cannot get additional information about the links in Alice's data. On the other hand, although Alice knows the true value of the received bit by decryption, she does not know the semantic meaning of the bit since a bit can be any link information in Bob's data, any bit in the binary representation of a similarity score between any two vertices, or even a temporary bit appearing only during the computation process. Consequently, the re-encryption protocol is efficient and satisfactory for our problem scenario.

²Given the public key, FHE [3] allows one to test the magnitude of noise in a ciphertext. If the noise grows beyond a certain ratio of the decryption radius, there is a demand of re-encryption.

³Refer to Footnote 2.

V. PERFORMANCE ANALYSIS

A. Security of PP-SimRank

The purpose of PP-SimRank is to achieve SimRank similarity score co-computation based on a joint virtual network $G = (V^A \cup V^B, U, E^A \cup E^B)$ while protecting every link information in both E^A and E^B from being stolen. In the analysis, we ask two questions: (1) Can Bob who has the whole network (in virtual) derive the links in Alice's data by generating many plaintext-ciphertext pairs with the given public key \mathbf{pk} ? (2) Can Alice or Bob derive the links in the other's data after the similarities between nodes in U is revealed?

The first question is also known as a birthday attack [8]. Since, in PP-SimRank, the public key \mathbf{pk} generated by Alice is shared with Bob, Bob is able to generate cipher-plaintext pairs by himself. The cipher-plaintext pairs he generated may contain a cipher that collides with an encrypted data received from Alice. The link information in E^A , therefore, has a chance to be stolen by Bob. However, according to the study [3], the ciphertext space of FHE is enormous large, i.e., ciphertext $c \in \mathbb{Z}_d = \{1, 2, \dots, d\}$ where $\log_2 d \in \{195764, 785006, 3148249\}$ w.r.t. the security level toy, small and median. Under the analysis of the birthday problem [8], the expected number of ciphertexts Bob has to generate before he finds the first collision is about $\sqrt{\frac{\pi}{2}}d$, which is still significantly large. Hence, the probability that Bob finds a collision and steals the link information from Alice is negligibly small.

For the second question, the answer is “No”. The reason is that, in addition to having the pairwise similarity scores $S(w_y)$ between nodes in U , a party also needs the pairwise similarity scores $S(w_x)$ between the nodes in $(V^A \cup V^B)$ so that the link information of the other party can be derived by solving linear algebra problems. However, at the end of PP-SimRank protocol, the similarity score $S(w_x)$ between the nodes in $(V^A \cup V^B)$ are not available for both Alice and Bob since Alice does not have the data and Bob does not have the secret key for decryption. Consequently, in a semi-honest security model, PP-SimRank does not give a chance to a party to steal the link information of the others.

B. Communication and computation complexity

We illustrate the communication and computational cost of PP-SimRank on a joint network $G = (V^A \cup V^B, U, E^A \cup E^B)$ in this section. Assume there are m nodes in $(V^A \cup V^B)$, n nodes in U , and the dimension of the node-pair graph matrix M is $C_2^m \times C_2^n$, i.e., $O(m^2) \times O(n^2)$. Let l be the number of bits used for binary representation of a SimRank score in plaintext, and k be the number of bits for a ciphertext transmission. First, we analyze the computational cost on the calculator, Bob. To construct the virtual node-pair graph G^2 in step 3 of PP-SimRank, Bob computes each element $\pi_{x,y}$ in M based on Equations (7), consisting of two additions $+$ and three multiplications \times . Therefore, Bob performs totally $O(m^2 n^2)$ additions and multiplications to fill the whole graph matrix. Next, in step 4 of PP-SimRank, Bob performs the SimRank score computation in ciphertext field according to Equations (10) and (11). For each score $\Theta(w_x)$ (or $\Theta(w_y)$), the required arithmetic operations on the cipher string include $O(1)$ binary division $\hat{\div}$, $O(n^2)$ (or $O(m^2)$) binary additions $\hat{+}$ and $O(n^2 + 1)$ (or $O(m^2 + 1)$) binary multiplications $\hat{\times}$, because the number of nodes in $(V^A \cup V^B)^2$ and U^2 are $O(m^2)$ and $O(n^2)$, respectively. We can

TABLE II
THE EXECUTION TIME (SEC) OF (BINARY) ARITHMETIC OPERATIONS IN CIPHERTEXT FIELD.

Arithmetic Operation	Security Level (dimension)		
	512	2048	8192
+	1.0×10^{-5}	3.0×10^{-5}	1.4×10^{-4}
\times	4.67×10^{-3}	0.03	0.135
$\hat{+}$	0.81	5.58	54.92
$\hat{-}$	1.11	8.61	92.7
$\hat{\times}$	5.30	47.06	584.00
$\hat{\div}$	22.39	191.94	1961.00

TABLE III
THE NUMBERS OF RE-ENCRYPTIONS INVOLVED DURING VARIOUS OPERATIONS.

Arithmetic Operation	Num. of Re-encryptions		
	max	Avg.	min
$+, \times, \hat{+}, \hat{-}$	0	0	0
$\hat{\times}$	40	36.04	33
$\hat{\div}$	280	224.31	191

further improve the performance by replacing the binary multiplication of $(\Theta(w) \hat{\times} \pi)$ in Equations (10) and (11) with l simple multiplications \times as explained in Section IV-B. The complexity for computing a $\Theta(w_x)$ (or $\Theta(w_y)$) therefore becomes $O(1)$ binary division $\hat{\div}$, $O(n^2)$ (or $O(m^2)$) binary additions $\hat{+}$, $O(n^2l)$ (or $O(m^2l)$) simple multiplications and $O(1)$ binary multiplication. The total binary arithmetic operations required in one iteration to compute all the SimRank scores are thus $O(m^2 + n^2)$ binary divisions, $O(m^2 + n^2)$ binary multiplications, $O(m^2n^2)$ binary additions and $O(m^2n^2l)$ simple multiplications. Table II shows the execution efficiency of each (binary) arithmetic operation in ciphertext field.

In PP-SimRank, the communication between Alice and Bob occurs in three steps, including encrypted link data transmission after step 2, resulting cipher SimRank score transmission after step 5 and re-encrypt protocol required during computation. First, assuming that Alice has $|V^A|$ nodes that exclusively owned by herself. She thus needs totally $(n|V^A|)$ bits to encode all the link information between V^A and U and transmits totally $O(kn|V^A|)$ bits data to Bob for subsequent calculations. Second, after completing the SimRank score computation, Bob will send $O(n^2)$ noisy cipher SimRank scores $\Theta'(w_y) = \Theta(w_y) \hat{+} r_y$ to Alice after step 5. Since each cipher score is represented by l ciphers and each cipher needs k bits for transmission, the communication cost is $O(n^2lk)$ bits. Finally, as shown in Table III, the times of re-encryption required vary with binary arithmetic operations on different ciphertexts. Here we evaluate the communication cost in one re-encryption protocol: Alice and Bob send one cipher to each other. Therefore, the cost is $O(k)$ bits.

VI. EXTENSION TO MULTI-PARTY SCENARIO

In this section, we further extend the PP-SimRank protocol to a more general case, i.e., multi-party scenario. We assume there are totally s parties and the i -th party holds an information network $G^i = (V^i, U, E^i)$ where the vertex sets V^1, V^2, \dots , and V^s are mutually disjoint. The purpose is to let all parties learn the similarity between nodes in U based on their joint network $G = (\cup V^i, U, \cup E^i)$ while the links held by each party are strictly known to itself.

Similar to the two-party scenario, here PP-SimRank also asks two of the s parties to act as a cryptographer (Alice) and a calculator (Bob) and to perform their duty respectively. For the rest parties, they encrypt their data with the public key \mathbf{pk} received from the cryptographer, and then transmit the encrypted data to the calculator after step 2 of PP-SimRank protocol (shown in Figure (3)). In addition, each of the rest parties also determines a random number list R^i containing C_2^n elements, encrypts each element r_y^i , $y = \{1, \dots, C_2^n\}$, with \mathbf{pk} , and sends the encrypted random numbers to the calculator. After the calculation of SimRank similarity is completed in step 4, the calculator adds each of the resulting SimRank scores $\Theta(w_y)$ with a set of random numbers $r_y^1, r_y^2, \dots, r_y^{s-1}$ collected from all parties except the cryptographer. Therefore, at the end, the cryptographer will hold the noisy scores $S'(w_y)$ after decryption, and each of the other parties will have a list of random numbers that have been added to the noisy scores $S'(w_y)$. The correct SimRank scores then can be derived after all parties exchange these information with each other.

Now consider the corresponding complexities. Assume there are totally m nodes in $(V^1 \cup V^2 \cup \dots \cup V^s)$, i.e., $m = |V^1| + |V^2| + \dots + |V^s|$ and n nodes in U , i.e., $n = |U|$. The dimension of the node-pair graph matrix M is therefore $O(m^2) \times O(n^2)$, which is the same with the dimension in the two-party scenario. As the result, the computational complexity of multi-party PP-SimRank is identical to that we have analyzed in the two-party scenario. On the other hand, the communication cost slightly increases. Assume that each score uses l bits for its binary representation and each ciphertext needs k bits for its transmission. There is a new cost $O((s-2)n^2lk)$ for transmitting $(s-2)$ encrypted lists of random numbers from all parties, except the cryptographer and the calculator, to the calculator, while the cost for encrypted link data transmission becomes $O((m - |V^B|)nk)$ bits. The cost for transmitting the resulting cipher SimRank scores remains the same.

VII. CONCLUSION

In this paper, we have addressed the problem of privacy-preserving co-computation of link-based similarity measure in a distributed bipartite network, and proposed a new privacy-preserving protocol, PP-SimRank, as a solution. PP-SimRank strictly protects each party's link information and reveals nothing but the desired similarity measures on the (virtual) joint network. We also implemented the basic binary arithmetic operations in ciphertext field to securely compute SimRank scores under fully-homomorphic encryption (FHE) and demonstrated the potential of FHE in privacy-preserving data mining.

REFERENCES

- [1] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Mining hidden community in heterogeneous social networks. In *Proceedings of the 3rd international workshop on Link discovery*, pages 58–65. ACM, 2005.
- [2] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.
- [3] C. Gentry and S. Halevi. Implementing gentrys fully-homomorphic encryption scheme. *Advances in Cryptology–EUROCRYPT 2011*, pages 129–148, 2011.
- [4] L. Getoor and C. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.
- [5] R. Jarvis and E. Patrick. Clustering using a similarity measure based on shared near neighbors. *Computers, IEEE Transactions on*, 100(11):1025–1034, 1973.
- [6] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.
- [7] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1298–1306. ACM, 2011.
- [8] J. Kemeny, J. Snell, and G. Thompson. *Introduction to finite mathematics*. Prentice-Hall, 1966.
- [9] Y. Koren, S. North, and C. Volinsky. Measuring and extracting proximity graphs in networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(3):12, 2007.
- [10] E. Leicht, P. Holme, and M. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.
- [11] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 465–476. ACM, 2010.
- [12] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.
- [13] A. Maguitman, F. Menczer, F. Erdinc, H. Roinestad, and A. Vespignani. Algorithmic computation and approximation of semantic similarity. *World Wide Web*, 9(4):431–456, 2006.
- [14] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 797–806. ACM, 2009.